IMPLEMENTASI STOP WORD REMOVAL UNTUK PEMBANGUNAN APPLIKASI ALKITAB BERBASIS WINDOWS 8

Alvin Setiawan
Erick Kurniawan, Wimmie Handiwidjojo

Abstrak

Pencarian adalah sebuah fitur utama yang sangat penting untuk ada dalam sebuah aplikasi. Terutama jika aplikasi tersebut akan berurusan dengan sumber data yang besar dan kurang teratur seperti data dalam konteks ayat Alkitab. Namun, dengan adanya fitur tersebut muncul juga beberapa masalah baru yaitu bagaimana menyediakan fitur pencarian yang memiliki akurasi dan performa yang baik. Oleh karena itu dalam penelitian ini, dikembangkan suatu aplikasi yang dapat menyediakan fitur pencarian dengan tingkat akurasi dan performa yang tinggi dengan menggunakan metode stopword removal. Dengan metode ini, kata-kata yang kurang begitu penting akan dihilangkan dari kata kunci pencarian sehingga diharapkan akan meningkatkan performa pencarian. Dan setelah dilakukan penelitian ini, dihasilkan sebuah aplikasi Alkitab dengan basis Windows 8 yang menyediakan fitur pencarian dengan menerapkan metode stopword removal..

Kata Kunci : *Alkitab, Text Mining, Stop Word Removal, Pencarian*

1. Pendahuluan

Pencarian adalah sebuah fitur utama yang sangat penting untuk ada dalam sebuah aplikasi. Terutama jika aplikasi tersebut akan berurusan dengan sumber data yang besar dan kurang teratur seperti data dalam konteks ayat Alkitab. Namun, dengan adanya fitur tersebut bukan berarti tidak ada masalah lainnya. Akan dibutuhkan waktu yang sangat lama jika pengguna awam yang tidak mengerti urutan atau aturan penyimpanan data tersebut ingin mencari sebuah data pada kumpulan data tersebut.

Selain itu, masalah lain yang sering dihadapi dalam melakukan pencarian adalah performa pencarian itu sendiri. Data yang sangat banyak dan tidak ter-index dengan baik akan menyulitkan sistem yang dibuat untuk melakukan pencarian karena akan mencari pada semua kolom dan setiap data yang ada untuk menemukan kata kunci yang dimasukkan oleh pengguna. Oleh karena itu akan sangat diharapkan pada setiap sistem atau aplikasi yang akan dibangun untuk menyediakan fitur pencarian yang memiliki performa yang baik dan tingkat akurasi yang tinggi..

2. Landasan Teori

a. Text Mining

Text mining menurut buku "*Text Mining Predictive Methods for Analyzing Unstructured Information*" (Wess, Indurkhya, Zhang, & Damerau, 2005, hal. 1-2), didefininisikan sebagai sebuah proses untuk menggali atau mendapatkan informasi dari kumpulan data-data yang tidak terstruktur sebagai contoh adalah sebuah text. Penggalian informasi ini bisa dikategorikan ke dalam berbagai cara, seperti di bawah ini:

1) *Information Retrieval*: menggunakan pencarian dokumen untuk mendapatkan informasi yang lebih akurat.
2) Klasifikasi dokumen: membagi dokumen ke dalam kelas-kelas yang etlah ditentukan sebelumnya. Misalnya secara otomatis dapat menentukan apakah dokumen ini masuk dalam kategori politik, ekonomi, militer dan lain sebagainya.

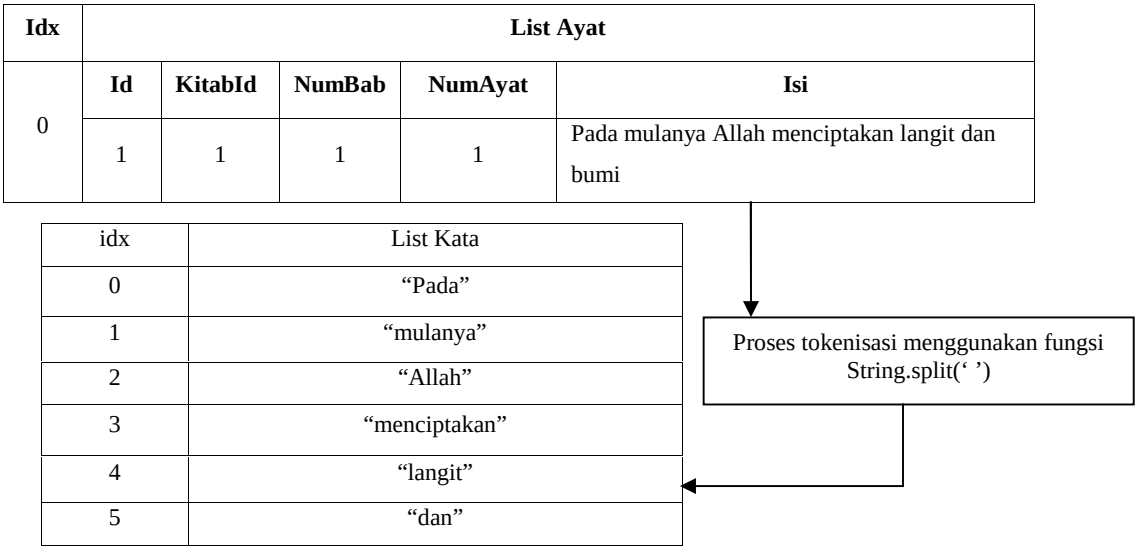
3) *Document clustering*: mirip dengan klasifikasi dokumen, hanya saja kelas dokumen tidak ditentukan sebelumnya. Misalnya berita tentang lalu lintas menjadi satu dengan berita tentang kriminal karena di dalamnya banyak memuat orang tewes, cedera, dan lain-lain.
4) Peringkasan teks: menghasilkan ringkasan suatu dokumen secara otomatis.
5) Ekstraksi informasi: mengekstrak informasi yang dianggap penting dari suatu dokumen. Misalnya pada dokumen lowongan, walaupun memiliki format beragam dapat diekstrak secara otomatis bagian-bagian yang diinginkan, seperti nama pekerjaan, tingkat pendidikan, lokasi, dan lain sebagainya.

Dalam penelitian ini, yang akan digunakan adalah sebagian kecil dari teori *text mining ini* yaitu bagian *information retrieval* atau pencarian data dengan menggunakan *stopword removal* dan melakukan parsing untuk menelusuri isi dari sebuah *file text* dan memasukkannya ke dalam sebuah database yang terstruktur dan membuat indentasi untuk kitab Mazmur.

b. Tokenizing

Tokenizing atau tokenisasi adalah proses yang paling awal dalam melakukan *text mining.* Dalam proses ini, *input stream* yang didapat dari *file text* akan dipecah-pecah menjadi bagian bagian yang lebih kecil. Sebagai contoh pemecahan kalimat menjadi kata-kata (*tokens*).

Dalam penelitian ini, proses tokenisasi dilakukan menggunakan bahasa pemrograman C#, dan data mentah yang digunakan dalam bentuk ".txt" dengan format *TSV* atau *Tab Separated Value.* Maka hal yang perlu dilakukan untuk melakukan tokenisasi ini adalah dengan membaca file teks dengan fungsi *streamreader.* Dan kemudian memproses setiap baris yang didapat dengan menggunakan fungsi string *split,* untuk mendapatkan *array* kata-kata dari sebuah kalimat. Berikut adalah contoh penerapannya:



Gambar 1. Proses Tokenisasi pada ayat Alkitab

Dari gambar di atas dapat dilihat bahwa yang pertama dilakukan adalah membaca baris teks dalam file teks yang dimiliki menggunakan fungsi *streamreader.* Kemudian dilakukan iterasi atau perulangan menggunakan *while loop* dengan kondisi selama baris pada file teks tersebut belum habis, maka setiap baris akan dibaca menggunakan fungsi *readline* dan dipecah berdasarkan karakter tab ("\t") untuk mendapatkan setiap kolom yang ada.

Hasil dari pemecahan kolom tersebut disimpan didalam sebuah *array of string* bernama "List Ayat" yang kemudian diambil kolom kelima yang berisikan detail dari ayat tersebut. Pada kolom kelima ini kemudian dilakukan proses *tokenizing* atau tokenisasi dengan menggunakan fungsi *string split* dengan *delimiter* atau pemisah berupa karakter spasi (" ") dan ditampung dalam sebuah *array of string* bernama "List Kata". Dengan demikian dapat diproses setiap katanya dengan menggunakan *foreach loop* untuk "List Kata".

c. Metode Stop Word Removal

Langkah yang selanjutnya adalah *Stop Word Removal. Stop Word* sendiri didefinisikan sebagai sekumpulan kata yang tidak berhubungan (*irrelevant*) dengan subyek utama yang dimaksud, meskipun kata tersebut sering muncul didalam data yang digunakan. Kata-kata yang dimaksud biasanya adalah jenis kata sambung, imbuhan, dan lain sebagainya.

Berikut adalah contoh algoritma yang digunakan untuk menghilangkan *stopwords* dari kata kunci pencarian yang dimasukkan oleh pengguna:

1) Memasukan daftar stopword dari database ke dalam array stoplist.
2) Menampung input kata kunci pencarian ke dalam variabel.
3) Memecah *variable string* menggunakan fungsi string split ke dalam array kata kunci.
4) Inisialisasi variabel ketemu berisi nilai Boolean *false.*
5) Apakah elemen pada array kata kunci sama dengan elemen pada stoplist? Jika *true*, lakukan langkah 6, jika tidak terpenuhi lakukan langkah 7.
6) Ubah nilai variabel ketemu menjadi *true.* Lakukan langkah 8.
7) Lakukan langkah 4-5 hingga seluruh elemen pada array stoplist habis.
8) Apakah variabel ketemu bernilai *false?* Jika kondisi terpenuhi, lakukan langkah 9, jika tidak terpenuhi lakukan langkah 10.
9) Masukan elemen array kata kunci yang dipilih ke dalam array hasil.
10) Lakukan langkah 4-9 hingga seluruh elemen pada array kata kunci habis.

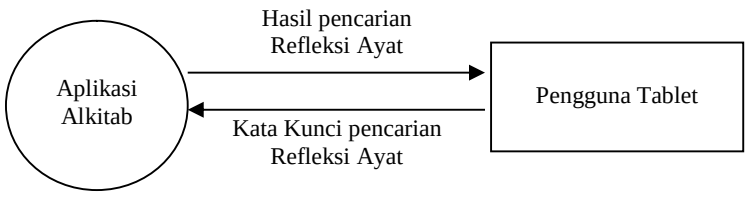
Dari algoritma di atas, akan dihasilkan sebuah array yang berisi kata kunci pencarian yang dimasukkan oleh pengguna namun sudah terbebas dari *stopword.* Sehingga hal berikutnya yang harus dilakukan adalah menyusun sebuah *SQL Query* dengan melakukan perulangan pada array hasil dan menyusun setiap kata kunci yang ada menjadi sebuah *SQL Query* yang lengkap seperti yang ditunjukkan pada gambar di bawah ini :

```
SELECT * FROM Ayat WHERE isi LIKE '%kata1%' AND isi LIKE '%kata2%' AND
isi LIKE '%kata3%';
```

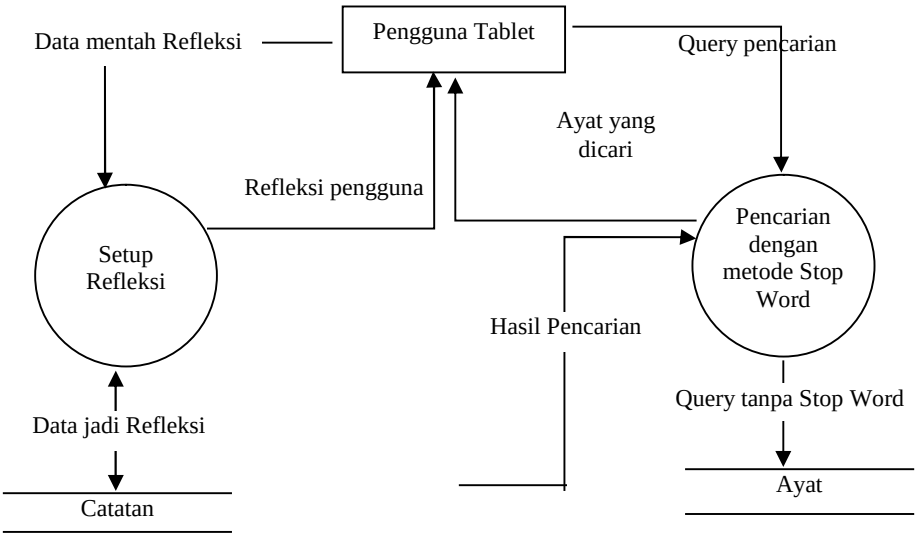
3. Perancangan Sistem

a. Data Flow Diagram

Diagram konteks adalah gambaran sistem yang paling umum. Diagram ini berfungsi untuk memperlihatkan interaksi sistem dengan lingkungan diluar sistem seperti entitas atau data eksternal. Berikut adalah diagram konteks untuk penelitian ini:



Gambar 2. Diagram Konteks

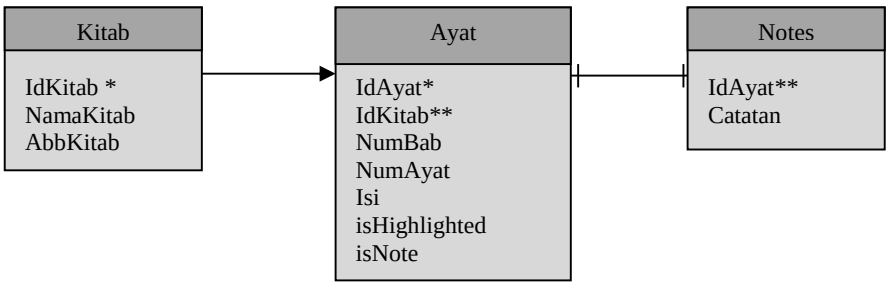


Gambar 3. DFD Level 1

Pada DFD level 1 di atas, dapat dilihat secara lengkap bahwa pengguna dapat berinteraksi dengan dua proses utama pada aplikasi Alkitab ini. Adapun kedua proses utama pada aplikasi Alkitab ini adalah *setup* refleksi pengguna dan pencarian. Dalam *setup* refleksi penguna, pengguna dapat meng-*input*-kan refleksinya ke dalam tabel catatan dan juga me-*retrieve* data-data refleksi yang sudah ada. Selain itu, pengguna juga dapat melakukan pencarian. Pengguna hanya perlu meng-*input*-kan kata kunci yang diinginkan, maka sistem secara otomatis akan memproses kata kunci tersebut dan membuang setiap kata yang terdaftar dalam *Stop List*, sehingga dihasilkan sebuah kata kunci pencarian yang terbebas dari *Stop Word*. Lalu sistem akan menampilkan hasil pencarian tadi kepada pengguna dalam bentuk daftar ayat yang mengandung kata kunci.

b. Normalisasi dan Domain Data

Entitas yang digunakan dalam penelitian ini adalah sebagai berikut:



Gambar 4. Entitas Database

Penjabaran secara lengkap dari aturan normalisasi dan validasinya dijelaskan di bawah ini:

1) *First Normal Form:* proses pertama dari normalisasi adalah memastikan bahwa tidak ada nilai ganda (*multi value*) pada tiap atribut yang ada. Dalam hal ini, setiap entitas yang ada sudah memenuhi *first normal form* ini, karena setiap atribut dalam entitas hanya bernilai tunggal (*atomic value*).

2) *Second Normal Form:* proses yang kedua dari normalisasi adalah memastikan bahwa tidak terjadi *Partial Dependency*, yaitu kondisi ketika salah satu atribut dalam satu entitas bergantung pada sebagian kunci primer yang ada di entitas tersebut. Dalam penelitian ini, bentuk kedua dari normalisasi ini dapat dilihat pada penjabaran di bawah ini :
   a) Kitab, PK (*Primary Key*): IdKitab, tidak mungkin terjadi *partial dependency*. Hal ini disebabkan karena IdKitab adalah kunci primer tunggal sehingga semua atribut yang ada sudah pasti ditentukan oleh kunci primer ini.
   b) Ayat, PK (*Primary Key*): IdAyat, tidak mungkin terjadi *partial dependency*. Hal ini disebabkan karena IdAyat adalah kunci primer tunggal sehingga semua atribut yang ada sudah pasti ditentukan oleh kunci primer ini.
3) *Third Normal Form:* proses yang ketiga dari normalisasi adalah memastikan tidak terjadi *Transitive Dependency*, yaitu kondisi ketika sebuah atribut dalam entitas tertentu ditentukan bukan oleh kunci primernya namun oleh atribut lain. Dalam penelitian ini, tidak mungkin terjadi *Transitive Dependency*, karena setiap atribut sudah bergantung penuh pada kunci primer mereka masing-masing.

Berikut ini adalah domain data:

1) Entitas Kitab

Tabel 1.
Domain Data Entitas Kitab

Nama Field	Property Data	Property Lainnya
IdKitab	Tipe : Integer (2) Format : 99 Rule : .not. EMPTY(IdKitab)	Arti : kode Kitab Unik : unik (kunci primer) Dukungan Kosong : tidak
NamaKitab	Tipe : Char (30) Format : - Rule : .not. EMPTY(NamaKitab)	Arti : nama Kitab Unik : tidak Dukungan Kosong : tidak
AbbKitab	Tipe : Char (20) Format : - Rule : .not. EMPTY(AbbKitab)	Arti : Singkatan nama Kitab Unik : tidak Dukungan Kosong : Tidak

2) Entitas Ayat

Tabel 2.
Domain Data Entitas Ayat

Nama Field	Property Data	Property Lainnya
IdAyat	Tipe : Integer (6) Format : 999999 Rule : .not. EMPTY(IdAyat)	Arti : kode Ayat Unik : unik (kunci primer) Dukungan Kosong : tidak
IdKitab	Tipe : Integer (2) Format : 99 Rule : .not. EMPTY(IdKitab)	Arti : kode Kitab Unik : tidak (kunci tamu) Dukungan Kosong : tidak
NumBab	Tipe : Integer (3) Format : 999 Rule : .not. EMPTY(NumBab)	Arti : nomor bab dari ayat Unik : tidak Dukungan Kosong : Tidak
NumAyat	Tipe : Integer (3) Format : 999 Rule : .not. EMPTY(NumAyat)	Arti : nomor urut ayat Unik : tidak Dukungan Kosong : Tidak
Isi	Tipe : Varchar (MAX) Format : - Rule : .not. EMPTY(Isi)	Arti : isi dari ayat Unik : tidak Dukungan Kosong : Tidak

isHighlighted	Tipe : Integer(1) Format : 9 Rule : .not. EMPTY(isHighlighted)	Arti : jenis warna stabilo Unik : tidak Dukungan Kosong : Tidak
isNote	Tipe : Integer (1) Format : 9 Rule : .not. EMPTY(isNote)	Arti : penanda adanya catatan Unik : tidak Dukungan Kosong : Tidak

3) Entitas Notes

Tabel 3.
Domain Data Entitas Notes

Nama Field	Property Data	Property Lainnya
IdAyat	Tipe : Integer (6) Format : 999999 Rule : .not. EMPTY(IdAyat)	Arti : kode Ayat Unik : tidak (kunci tamu) Dukungan Kosong : tidak
Catatan	Tipe : Varchar (MAX) Format : - Rule : .not. EMPTY(Catatan)	Arti : catatan dari pengguna untuk ayat tertentu Unik : tidak Dukungan Kosong : Tidak

4. Implementasi dan Analisa Sistem

a. Implementasi Sistem

Implementasi dari aplikasi yang dibangun dapat dikategorikan ke dalam tiga proses yaitu proses setup catatan, proses navigasi dan proses pencarian. Berikut adalah penjelasannya:

Gambar 5. Setup Catatan

1) Proses setup catatan

Proses ini dapat dilakukan langsung dari halaman utama dengan cara melakukan seleksi pada salah satu ayat yang diinginkan, maka akan muncul sebuah pop-up form untuk mengisi catatan. Setelah pengguna selesai memasukan catatan yang diinginkan, tekan tombol simpan dan data catatan akan tersimpan secara otomatis.

Untuk melakukan perubahan catatan, pengguna hanya perlu melakukan seleksi pada ayat yang diinginkan kemudian tekan tombol “lihat catatan”. Dan setelah jendela yang berisikan catatan muncul, lakukan perubahan yang diinginkan dan tekan tombol simpan. Dengan demikian perubahan tersebut akan tersimpan.

Untuk melakukan penghapusan catatan, pengguna dapat melakukan seleksi ayat yang diinginkan kemudian menekan tombol “lihat catatan”. Setelah itu hapus semua catatan yang telah ditulis dan tekan tombol “simpan”. Berikut adalah kode program untuk melakukan setup catatan:

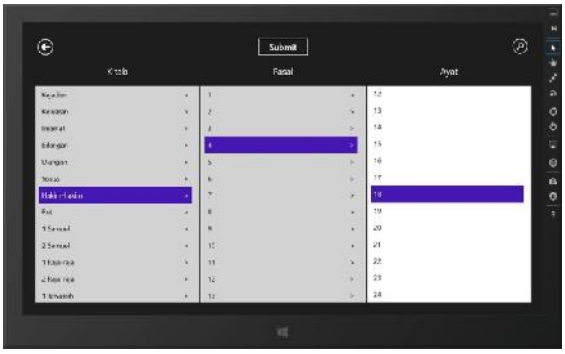
```
tamp.AyatId = selectedAyat.Id;
tamp.catatan = teksCatatan;
tp.insertCatatan(tamp);
tp.updateCatatan(tamp);
tp.deleteCatatan(tamp);
```

A

B

Kode program diatas menunjukan proses setup catatan. Secara garis besar kode tersebut dapat diuraikan sebagai berikut, ketika pengguna menekan tombol simpan, maka sistem akan mengambil IdAyat yang sedang dipilih dan isi catatan seperti pada bagian A, kemudian lakukan salah satu baris pada bagian B sesuai dengan kondisi yang terpenuhi, jika baru maka "*insert*", jika terjadi perubahan isi maka "*update*" dan terakhir jika isi catatan dikosongi maka akan di panggil metode "*delete*".

2) Proses Navigasi



Gambar 6. Proses Navigasi

Proses navigasi dapat diakses dengan cara menekan tombol yang bertuliskan judul kitab pada halaman utama. Maka pengguna akan diarahkan pada halaman seperti pada gambar di samping. Pengguna akan dihadapkan pada tiga buah kolom untuk memilih kitab, pasal dan bab yang diinginkan. Setelah pengguna menentukannya, tekan tombol "submit" untuk melihat ayat yang dituju pada halaman utama. Berikut adalah kode program untuk melakukan navigasi:

```
Kitab kitab = tp.getAllKitab().First(x => x.Id.Equals(ayatTerpilih.KitabId));
   t_Ayat      selectedItem      =      myDataModel.AyatGroupedDataSource.First(x      =>
x.namaKitab.Equals(kitab.namaKitab) && x.numBab.Equals(ayatTerpilih.numBab));
   this.flipViewAyat.SelectedItem = selectedItem;
this.flipViewAyatPotrait.SelectedItem = selectedItem;
```

Secara garis besar, baris program diatas dapat diuraikan sebagai berikut ini. Ketika pengguna telah menentukan ayat mana yang dituju, sistem akan mengambil data kitab yang terpilih untuk kemudian digunakan dalam pencarian ke koleksi ayat yang digunakan pada halaman utama. Setelah itu ketika ayat telah didapatkan, sistem akan menggunakan ayat tersebut untuk mengubah index dari *flipview control* pada halaman utama sesuai dengan ayat yang dipilih.

3) Proses Pencarian

Gambar 7. Proses Pencarian

Halaman pencarian dapat diakses dari halaman utama dengan menekan *icon* bergambar kaca pembesar pada bagian pojok kanan atas layar. Fasilitas pencarian ini menerapkan metode *Stopword Removal* dalam melakukan pencarian untuk menghasilkan pencarian yang lebih akurat dan efisien. Berikut adalah kode program untuk melakukan pencarian:

```
string[] arrKataKunci = kataKunci.Split(' ');
foreach (string kata in arrKataKunci) { bool ketemu = false;
foreach (Stopword item in stopList) {
if (kata.ToUpper() == item.stopword.ToUpper()) { ketemu = true;  break; }
}
if (!ketemu) { lstKtKunciTnpStopWord.Add(kata); }
}
```

Secara garis besar yang dilakukan dalam pencarian adalah memecah katakunci yang dimasukkan oleh pengguna berdasarkan spasi sehingga terpecah menjadi kata-kata tersendiri. Kemudian melakukan perulangan dalam kata-kata tersebut dan melakukan pengecekan terhadap koleksi stopword, jika ternyata ditemukan kata tersebut dalam koleksi stopword maka kata tersebut akan dihilangkan dari daftar kata kunci dan tidak diproses lebih lanjut.

b. Analisa Sistem

Dari penelitian ini, penyusun melakukan analisis terhadap metode *stopword removal* yang telah diterapkan dengan tujuan untuk mengetahui apakah metode tersebut benar-benar telah meningkatkan akurasi dan efisiensi waktu dari proses pencarian dalam aplikasi yang dibangun. Analisis tersebut dilakukan dengan cara melakukan uji coba sebagai berikut:

Tabel 4.
Hasil Uji Percobaan Pencarian

Nats Alkitab	Kata Kunci	Metode	Kata yang Dihilangkan	Jumlah Ayat	Waktu Dipakai
Yer 29:11	Sebab aku mengetahui rancangan	Ya	sebab, aku, mengetahui	64	109,72 ms
		Tidak	-	1	108,78 ms
	Rancangan damai sejahtera bukan rancangan kecelakaan	Ya	bukan	1	111,78 ms
		Tidak	-	1	101,84 ms
Yoh 3:16	Karena begitu besar kasih Allah akan dunia ini	Ya	karena, begitu, ini, besar, akan	9	129,48 ms
		Tidak	-	1	154,06 ms
	Supaya setiap orang yang percaya tidak binasa	Ya	supaya, setiap, orang, yang, tidak	6	98,06 ms
		Tidak	-	1	91,46 ms
Flp 4:13	Segala perkara dapat kutanggung	Ya	segala, dapat	1	121,47 ms
		Tidak	-	1	91,28 ms
	Dalam dia yang memberi	Ya	dalam, dia, yang, memberi	250	105,82 ms

	kekuatan	Tidak	-	1	94,54 ms
Rom 8:28	Allah turut bekerja dalam segala sesuatu	Ya	turut, segala, dalam, bekerja, sesuatu	4308	150,62 ms
		Tidak	-	1	130,09 ms
	Untuk mendatangkan kebaikan bagi yang mengasihi dia	Ya	untuk, mendatangkan, bagi, yang, dia	1	98,67 ms
		Tidak	-	1	105,57 ms
Ams 3:5	Percayalah kepada Tuhan dengan segenap hati	Ya	kepada, dengan	1	108,00 ms
		Tidak	-	1	97,37 ms
	Janganlah bersandar kepada pengertianmu sendiri	Ya	janganlah, kepada, sendiri	1	112,72 ms
		Tidak	-	1	101,04 ms

Dari hasil uji percobaan diatas, maka dapat dilakukan analisis untuk melihat keakuratan dan efisiensi dari metode yang digunakan. Berikut adalah hasil analisis uji percobaan tersebut:

Tabel 5.
Hasil Analisis Percobaan

Uji ke -	Kata kunci ke-	Metode	% akurasi		Selisih Waktu (metode – tanpa metode)	
1	1	Ya	1 : 64 = 1,56 %		0,94 ms	
		Tidak	1 : 1 = 100 %			
	2	Ya	1 : 1 = 100 %		9,94 ms	
		Tidak	1 : 1 = 100 %			
Sub kesimpulan		Ya	Avg akurasi	50,78%	Avg Waktu	110,75 ms
		Tidak		100 %		105,31 ms
2	1	Ya	1 : 9 = 11,11 %		-24,58 ms	
		Tidak	1 : 1 = 100 %			
	2	Ya	1 : 6 = 16,67 %		6,6 ms	
		Tidak	1 : 1 = 100 %			
Sub kesimpulan		Ya	Avg akurasi	19,45%	Avg Waktu	113,77 ms
		Tidak		100 %		122,76 ms
3	1	Ya	1 : 1 = 100 %		30,19 ms	
		Tidak	1 : 1 = 100 %			
	2	Ya	1 : 250 = 0,4 %		11,28 ms	
		Tidak	1 : 1 = 100 %			
Sub kesimpulan		Ya	Avg akurasi	50,2%	Avg Waktu	113,65 ms
		Tidak		100 %		92,91 ms
4	1	Ya	1 : 4308 = 0,02%		20,53 ms	
		Tidak	1 : 1 = 100 %			
	2	Ya	1 : 1 = 100 %		-6,9 ms	
		Tidak	1 : 1 = 100 %			
Sub kesimpulan		Ya	Avg akurasi	50,01%	Avg Waktu	124,65 ms
		Tidak		100 %		117,83 ms
5	1	Ya	1 : 1 = 100 %		10,63 ms	
		Tidak	1 : 1 = 100 %			
	2	Ya	1 : 1 = 100 %		11,68 ms	
		Tidak	1 : 1 = 100 %			
Sub kesimpulan		Ya	Avg akurasi	100 %	Avg Waktu	110,36 ms
		Tidak		100 %		99,21 ms
Kesimpulan		Ya	Avg akurasi	52,97%	Avg Waktu	114,63 ms
		Tidak		100 %		107,60 ms

Dari data analisis di atas dapat dilihat bahwa dari sudut pandang akurasi hasil pencarian, pencarian tanpa menggunakan metode selalu menghasilkan persentase akurasi 100%, hal ini

disebabkan karena semakin banyak kata kunci yang digunakan maka pencarian akan semakin mengerucut. Selain itu, faktor lain yang mendukung terjadinya hal tersebut adalah pencarian dilakukan dengan memeriksa kandungan kata perkata sehingga tidak dibutuhkan kata kunci yang benar-benar pasti, hanya dengan kata dasar yang sama ayat akan tetap ditemukan. Jika pencarian dilakukan dengan mencari kata kunci secara utuh sebagai satu kesatuan, maka kemungkinan besar pengguna tidak akan menemukan ayat yang diinginkan karena dibutuhkan imbuhan dan posisi kata yang benar-benar tepat seperti yang ada pada ayat yang dimaksud.

Sedangkan untuk pencarian dengan menggunakan metode *stopword removal*, terkadang akan menghasilkan hasil pencarian yang sangat majemuk. Hal ini disebabkan karena banyaknya kata yang dibuang dari kata kunci, dan juga faktor jumlah kandungan kata kunci yang tersisa pada Alkitab. Kesimpulan ini dibuktikan oleh percobaan keempat, pada pencarian di dalam nats Alkitab Roma 8:28. Pada pencarian menggunakan kata kunci pertama. Dari enam kata kunci yang digunakan, lima diantara termasuk dalam daftar *stopword* dan dihilangkan, sehingga hanya menyisakan kata “Allah” dalam kata kunci. Kata “Allah” tentu saja akan sangat banyak ditemui di dalam Alkitab karena merupakan pembahasan utama dalam Alkitab, baik Perjanjian Lama ataupun Perjanjian Baru sehingga pada percobaan ini menghasilkan lebih dari 4000 ayat lain yang sebenarnya sangat jauh dari kata kunci yang dimasukkan.

Untuk segi efisiensi waktu yang dibutuhkan ketika mengeksekusi perintah SQL yang dihasilkan. Dari seluruh percobaan yang dilakukan, dapat disimpulkan bahwa sebenarnya pencarian dengan menggunakan metode dan tanpa menggunakan metode tidak memiliki perbedaan yang signifikan dalam hal waktu eksekusi perintah SQL. Hal ini dapat dilihat dari selisih rata-rata waktu antara dengan metode dan tanpa metode dari seluruh percobaan yang dilakukan hanya terpaut 7,03 ms. Terlebih satuan yang digunakan adalah *milisecond* (ms), atau seperseribu detik, tentu saja bagi pengguna selisih tersebut tidak akan ada bedanya.

Sedangkan dari hasil percobaan diatas, faktor yang paling mempengaruhi waktu yang dibutuhkan untuk eksekusi program adalah ke-khusus-an kata kunci yang digunakan dalam konteks data. Hal ini dibuktikan pada percobaan ketiga untuk nats Alkitab Yohanes 3:16 pada bagian pencarian tanpa menggunakan metode dan pada percobaan keempat untuk nats Roma 8:28 pada bagian pencarian menggunakan metode. Waktu yang dibutuhkan untuk mengeksekusi kedua perintah adalah 154,06 ms dan 150,62 ms, ini adalah waktu terlama dari seluruh percobaan yang dilakukan. Hal ini disebabkan karena pada kata kunci pencarian terdapat kata “Allah”, “kasih”, dan “dunia”. Kata-kata tersebut sangat umum ditemukan dalam Alkitab sehingga eksekusi perintah SQL membutuhkan waktu yang lebih lama.

5. Penutup

Dari penelitian yang telah dilakukan dengan menggunakan metode *stopword removal* ini, maka dapat diambil beberapa kesimpulan, diantaranya adalah:

a. Metode *Stop Word Removal* tidak cocok untuk diterapkan dalam pencarian dengan konteks data seperti Alkitab, karena metode tersebut akan mengaburkan atau mengurangi akurasi hasil pencarian.
b. Metode *Stop Word Removal* tidak menambah efisiensi waktu yang dibutuhkan untuk mengeksekusi perintah SQL dalam pencarian dengan konteks data Alkitab.

Daftar Pustaka

Al-Shalabi, Riyadh, & Kanaan, Ghasan, & Jaam, Jihad M., & Hasnah, Ahmad, & Hilat, Eyat. *Stop Word Removal Algorithm for Arabic Language*. http://www.cs.wayne.edu/~eyad/sw_algo_arabic_2004.pdf. Diakses 21 Mei 2013.

Freeman, Adam. 2012. *Metro Revealed: Building Windows 8 Apps with XAML and C#*. New York. Springer Science.

Pramudya, Puja. 2013. *Membuat Aplikasi untuk Windows 8*. http://windows8applications.codeplex.com/. Diakses pada tanggal 1 February 2013.

Tabor, Bob. Windows 8 Camp: Introduction to Building Metro Apps – Using C#, XAML & WINRT. channel9.msdn.com/Blogs/bursteg/Part-4-Windows-8-Camp-Introduction-to-Building-Metro-Apps-Part-4. Diakses pada tanggal 20 February 2013.

Weiss, Sholom M., & Indurkhya, Nitin, & Zhang, Tong, & Damerau, Fred J. *Text Mining: Predictive Method for Analyzing Unstructured Information*. United States of America. Springer Science.

Wibawa, Made Satria. 2012. *Pengembangan Aplikasi Web Based Documents Similarity Measure Menggunakan Model Ruang Vektor pada Dokumen Berbahasa Indonesia*. http://www.pti-undiksha.com/karmapati/vol1no3/6.pdf. Diakses pada tanggal 29 November 2012.